

Regular Expression

1) Regular Expression Basics

Regular expressions are made up of normal characters and *metacharacters*. Normal characters include upper and lower case letters and digits. The metacharacters have special meanings and are described in detail below. Regular expressions are case sensitive.

In the simplest case, a regular expression looks like a standard search string. For example, the regular expression "testing" contains no metacharacters. It will match "testing" and "123testing" but it will not match "Testing". To really make good use of regular expressions it is critical to understand metacharacters. The table below lists metacharacters and a short explanation of their meaning.

Metacharacter Description

.	Matches any single character. For example the regular expression r.t would match the strings <i>rat</i> , <i>rut</i> , <i>rt</i> , but not <i>root</i> .
\$	Matches the end of a line. For example, the regular expression weasel\$ would match the end of the string " <i>He's a weasel</i> " but not the string " <i>They are a bunch of weasels.</i> "
^	Matches the beginning of a line. For example, the regular expression ^When in would match the beginning of the string " <i>When in the course of human events</i> " but would not match " <i>What and When in the</i> ".
*	Matches zero or more occurrences of the character immediately preceding. For example, the regular expression .* means match any number of any characters.
\	This is the quoting character, use it to treat the following character as an ordinary character. For example, \\$ is used to match the dollar sign character (\$) rather than the end of a line. Similarly, the expression \. is used to match the period character rather than any single character.
[]	Matches any one of the characters between the brackets. For example, the regular expression r[aou]t matches <i>rat</i> , <i>rot</i> , and <i>rut</i> , but not <i>ret</i> . Ranges of characters can be specified by using a hyphen. For example, the regular expression [c1-c2]
[^c1-c2]	[0-9] means match any digit. Multiple ranges can be specified as well. The regular expression [A-Za-z] means match any upper or lower case letter. To match any character <i>except</i> those in the range, the complement range, use the caret as the first character after the opening bracket. For example, the expression [^269A-Z] will match any characters except 2, 6, 9, and upper case letters.
	Or two conditions together. For example (him her) matches the line " <i>it belongs to him</i> " and matches the line " <i>it belongs to her</i> " but does not match the line " <i>it belongs to them.</i> " NOTE: this metacharacter is not supported by all applications.
+	Matches one or more occurrences of the character or regular expression immediately preceding. For example, the regular expression 9+ matches 9, 99, 999. NOTE: this metacharacter is not supported by all applications.
?	Matches 0 or 1 occurrence of the character or regular expression immediately preceding. NOTE: this metacharacter is not supported by all applications.
\{ ^ }	Match a specific number of instances or instances within a range of the preceding character. For example, the expression A[0-9]\{3\} will match "A" followed by exactly 3 digits. That is, it will match A123 but not A1234. The expression \{i,j\}
\{ i,j \}	[0-9]\{4,6\} any sequence of 4, 5, or 6 digits. NOTE: this metacharacter is not supported by all applications.
\(\)	Treat the expression between \ (and \) as a group. Also, saves the characters matched by the expression into temporary holding areas. Up to nine pattern matches can be saved in a single regular expression. They can be referenced as \1 through \9 .

2) Simple Examples

Here are a few representative, simple examples.

<i>vi command</i>	<i>What it does</i>
<code>:%s/ */ /g</code>	Change 1 or more spaces into a single space.
<code>:%s/ *\$/</code>	Remove all spaces from the end of the line.
<code>:%s/^ /</code>	Insert a space at the beginning of every line.
<code>:%s/^[0-9][0-9]* //</code>	Remove all numbers at the beginning of a line.
<code>:%s/[aeio]g/bug/g</code>	Change all occurrences of <i>bag</i> , <i>beg</i> , <i>big</i> , and <i>bog</i> , to <i>bug</i> .

3) Medium-level Example

Question: Change all instances of `foo(a,b,c)` to `foo(b,a,c)`. where a, b, and c can be any parameters supplied to `foo()`. That is, we must be able to make changes like the following:

Before	After
<code>foo(10,7,2)</code>	<code>foo(7,10,2)</code>
<code>foo(x+13,y-2,10)</code>	<code>foo(y-2,x+13,10)</code>
<code>foo(bar(8), x+y+z, 5)</code>	<code>foo(x+y+z, bar(8), 5)</code>

The following substitution command will do the trick :

```
:%s/foo(\([^,]*\),\([^,]*\),\([^,]*\))/foo(\2,\1,\3)/g
```

4) Regular Expressions In Various Tools

grep

grep is a program used to match regular expressions in one or more specified files or in an input stream. Its name programming language which can be used to perform data manipulation on files or pipes. For complete details, see the man page [grep\(1\)](#). Its peculiar name stems from its roots as a command in vi, *g/re/p* meaning global regular expression print.

For the examples below, assume we have the text below in a file named phone.txt. Its format is last name followed by a comma, first name followed by a tab, then a phone number.

```
Francis, John      5-3871
Wong, Fred         4-4123
Jones, Thomas     1-4122
Salazar, Richard  5-2522
```

<i>grep command</i>	<i>Description</i>
<code>grep '\t5-...1' phone.txt</code>	print all the lines in phone.txt where the phone number begins with 5 and ends with 1. Note that the tab character is represented by <code>\t</code> .
<code>grep '^S[^]* R' phone.txt</code>	print lines where the last name begins with S and first name begins with R.
<code>grep '^[JW]' phone.txt</code>	print lines where the last name begins with J or W

grep ',\t' phone.txt print lines where the first name is 4 characters. The tab character is represented by \t.
 grep -v '^[JW]' phone.txt print lines that do not begin with J or W
 grep '^[M-Z]' phone.txt print lines where the last name begins with any letter from M to Z.
 grep '^[M-Z].[12]' phone.txt print lines where the last name begins with a letter from M to Z and where the phone number ends with a 1 or 2.

egrep

egrep is an extended version of grep. It supports a few more metacharacters in its regular expressions. For the examples below, assume we have the text below in a file named phone.txt. Its format is last name followed by a comma, first name followed by a tab, then a phone number.

Francis, John 5-3871
 Wong, Fred 4-4123
 Jones, Thomas 1-4122
 Salazar, Richard 5-2522

<i>egrep command</i>	<i>Description</i>
egrep '(John Fred)' phone.txt	print all lines that contain the name <i>John</i> or <i>Fred</i> .
egrep 'John 22\$ ^W' phone.txt	print lines that contain <i>John</i> or that end with 22 or that begin with W.
egrep 'net(work)?s' report.txt	print lines in report.txt contain <i>networks</i> or <i>nets</i> .

Reference:

<http://www.zytrax.com/tech/web/regex.htm#experiment>

Exercises:

- 1) which could not be matched by regular expression ca*t
 a) ct b) cat c)caaaaaat d)ca*t

- 2) which could not be matched by regular expression ca+t
 a) ct b) cat c)caaaaaat

- 3) which could not be matched by regular expression ca?t
 a) ct b) cat c)caaaaaat

- 4) which could not be matched by regular expression ca\{2, 5\}t
 a) caat b) cat c)caaaaaat

- 5) Write up the regular expression for checking the validation of length-8 password which should include digitals and letters.

- 6) Given a file which includes the contact information of all your friends, try to filter out email address information. Contact information is composed of name, phone number, university, home address, website and email address. Every item is put in a line. Hint: The email address line describes a series of letters, digits, dots, underscores, percentage signs and hyphens, followed by an at sign, followed by another series of letters, digits and hyphens, finally followed by a single dot and between two and four letters.